Improving App Inventor Debugging Support

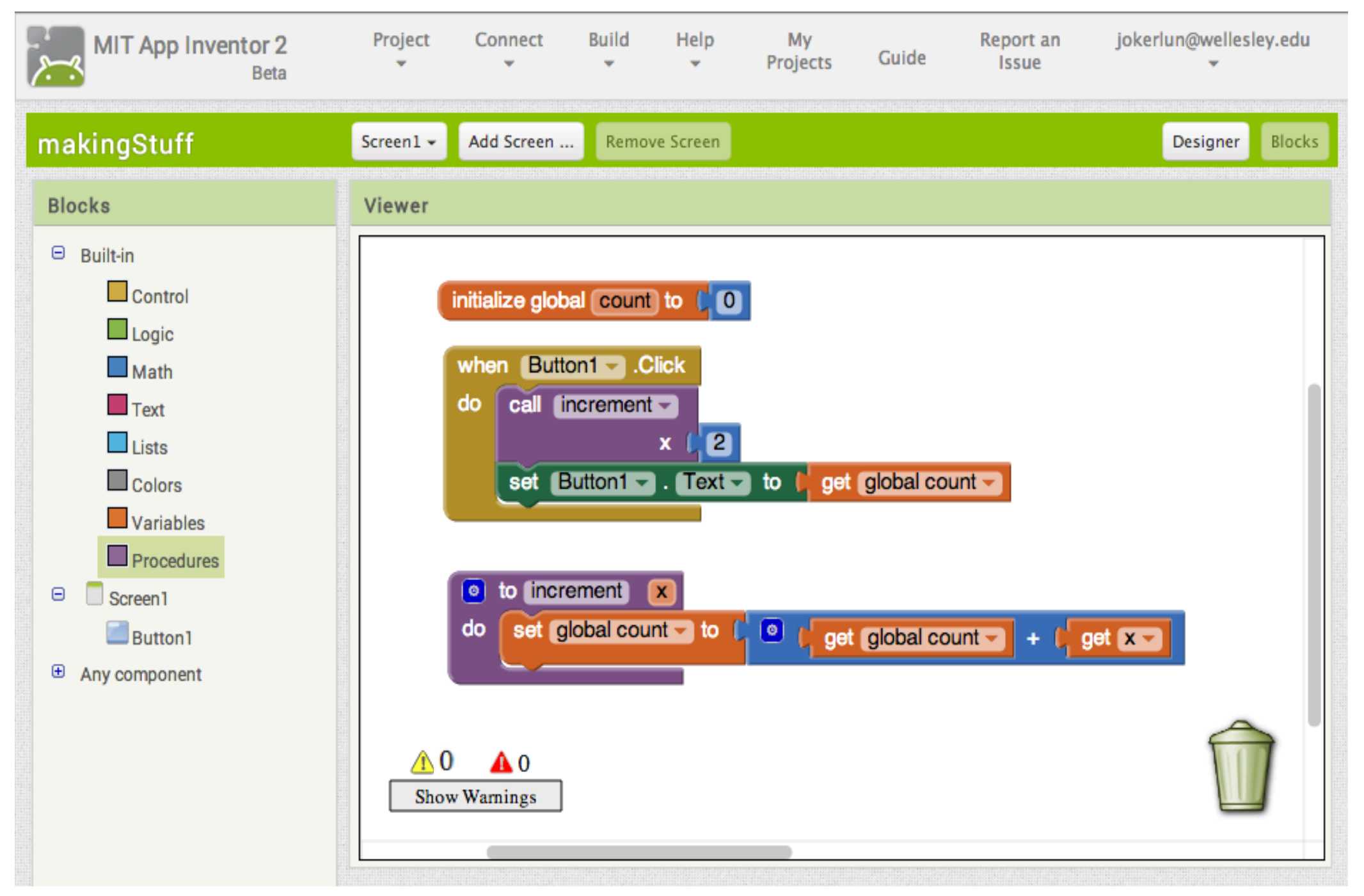
Johanna Okerlund, Wellesley College
Advisor: Franklyn Turbak

Project Goals

MIT App Inventor is a visual programming environment for creating apps for Android mobile devices. My project is to design and implement improved debugging support for App Inventor. With my changes, (1) error messages are now displayed on the block that is causing the error. I am also (2) designing and implementing a "watch" feature that allows users to track how the value of expressions change over time and (3) setting up the means to collect more data on runtime errors to better understand what additional support users might need.

App Inventor Background

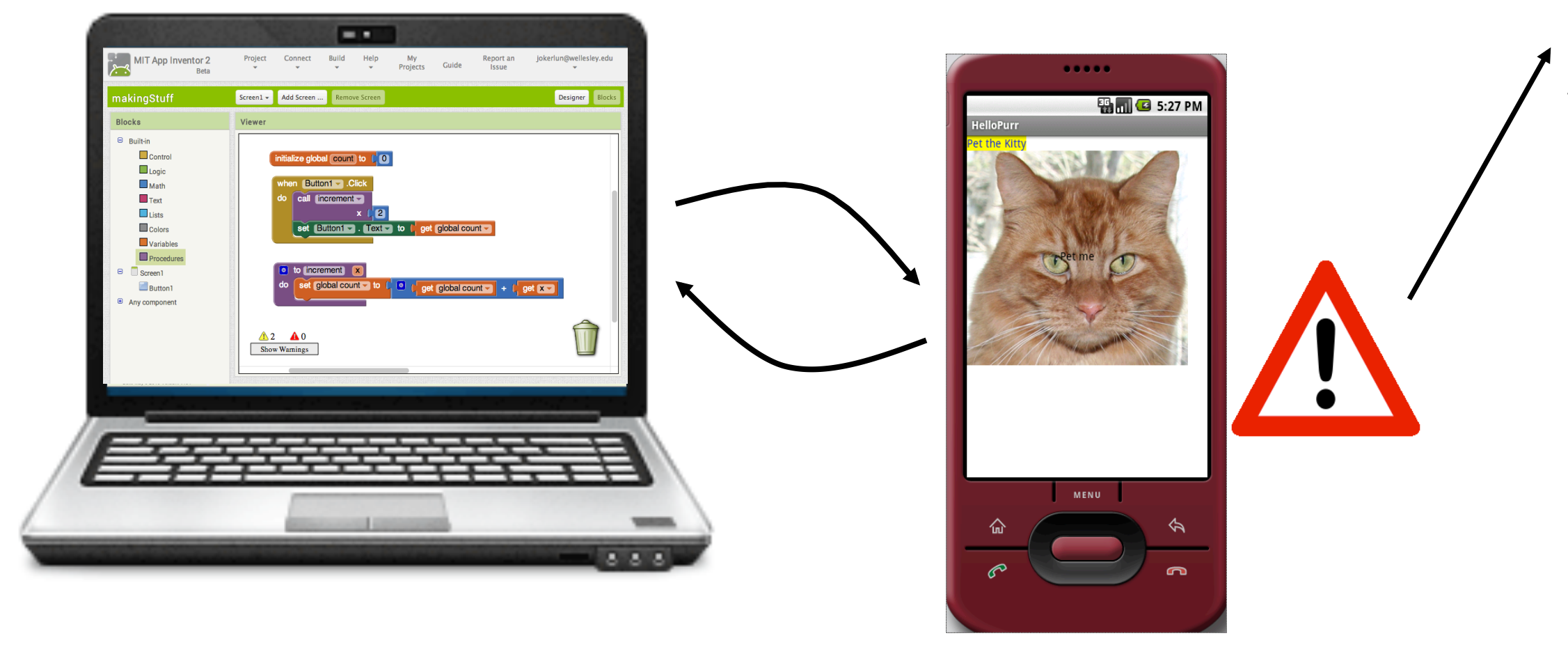
An App Inventor app is specified by a project consisting of a set of user interface components and a program that describes the behavior of these components. The program is created by connecting blocks resembling jigsaw puzzle pieces. Blocks languages like App Inventor and Scratch lower barriers for novices by eliminating or reducing many common programming errors and by providing visual guidance for choosing, assembling, and understanding program structures.



Sample App Inventor blocks program

Live Development Mode

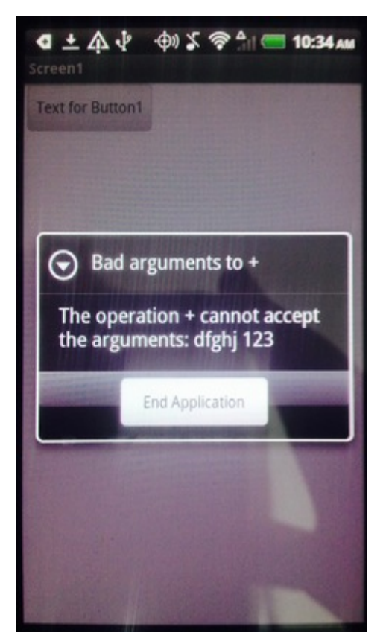
While programming, the user is in live development mode, where communication between the blocks program in the browser and the device is constant. The blocks editor converts the changed blocks to executable code to the phone and sends it over. The phone runs the new code and sends back status updates and responses to requests.



When a runtime error occurs on the phone, the error message is sent to a CouchDB database

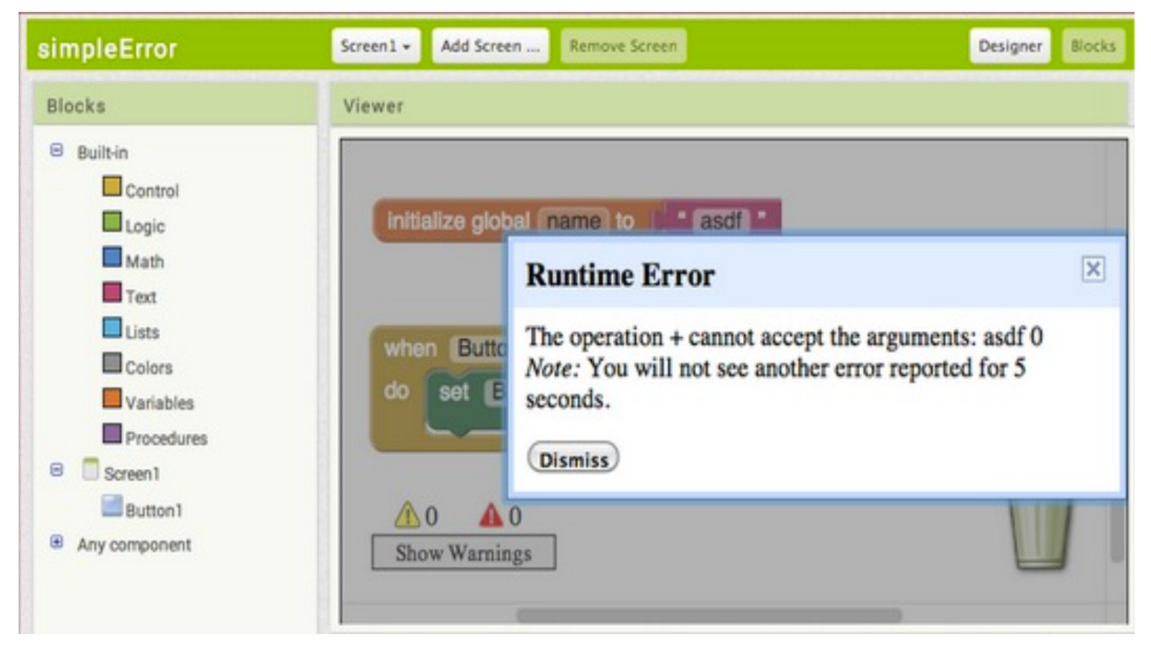
Previous work that included the analysis of the collected error reports shows that many users get errors and many of these people get the same or similar error messages repeatedly. This work lead to the conclusion that debugging support is needed and prompted my current work.

Errors on Blocks



Bad arguments to +

The operation + cannot accept the arguments: dfghj 123

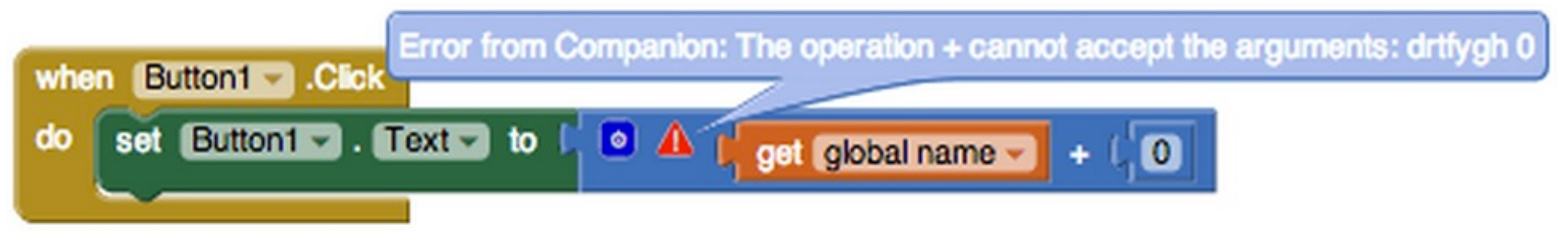


Runtime Error

The operation + cannot accept the arguments: asdf 0
Note: You will not see another error reported for 5 seconds.

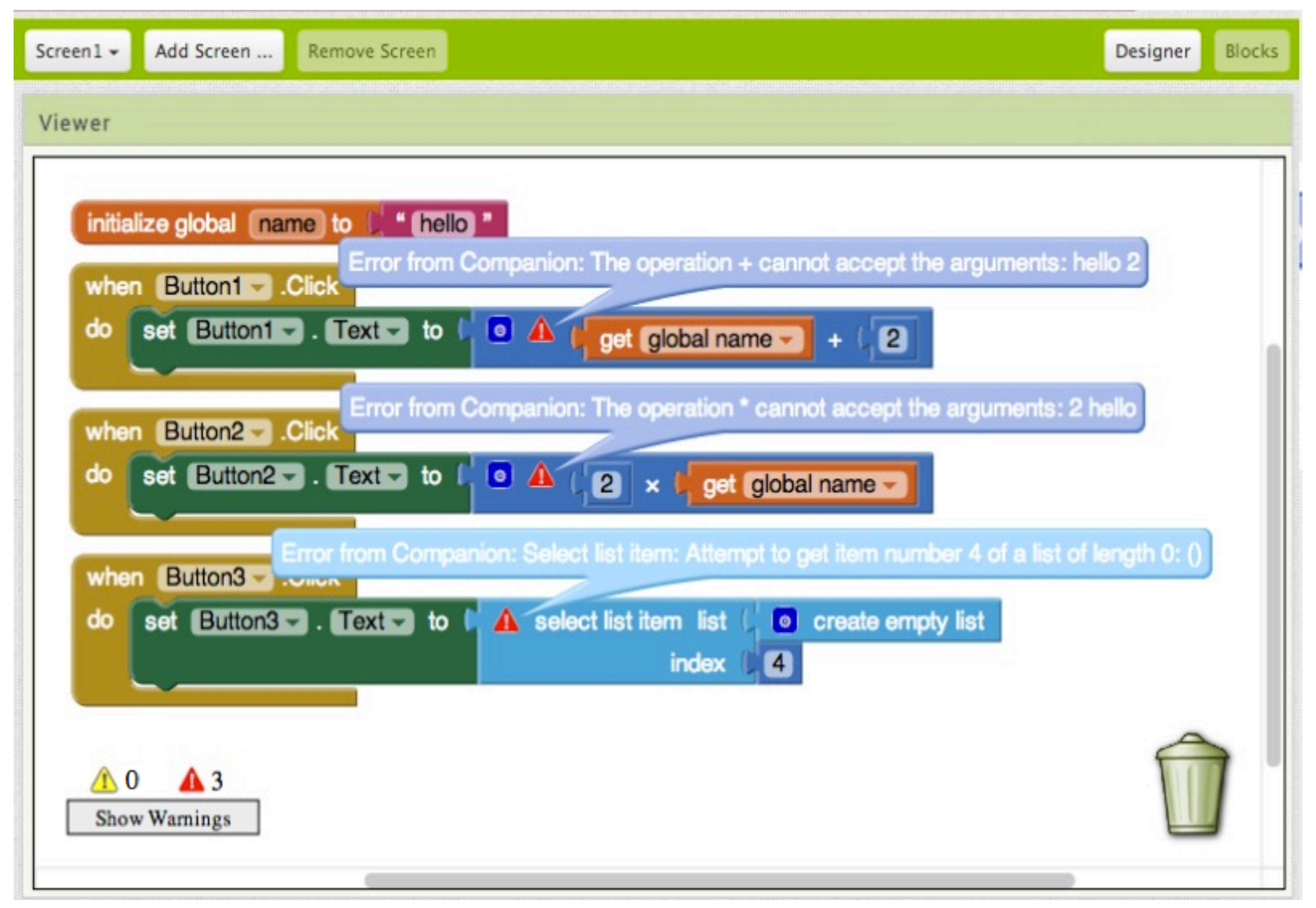
Although blocks eliminate many syntactic errors, runtime errors are still possible, such as type errors and index-out-of-bounds errors. Previously, when a program generated a runtime error, the message would appear on the device and in a dismissible dialogue box in the browser. However, this does not help users find the source of the error and doesn't allow them to look at the error and the code at the same time.

My solution is to display the error message to the block causing the error. Now the user knows exactly where the error is coming from and can refer back to the error message as they try to fix it.



Error from Companion: The operation + cannot accept the arguments: drtfygh 0

Multiple errors can also be viewed at the same time. When a block that previously generated an error executes without error, the message is removed from the block.



Error from Companion: The operation + cannot accept the arguments: hello 2

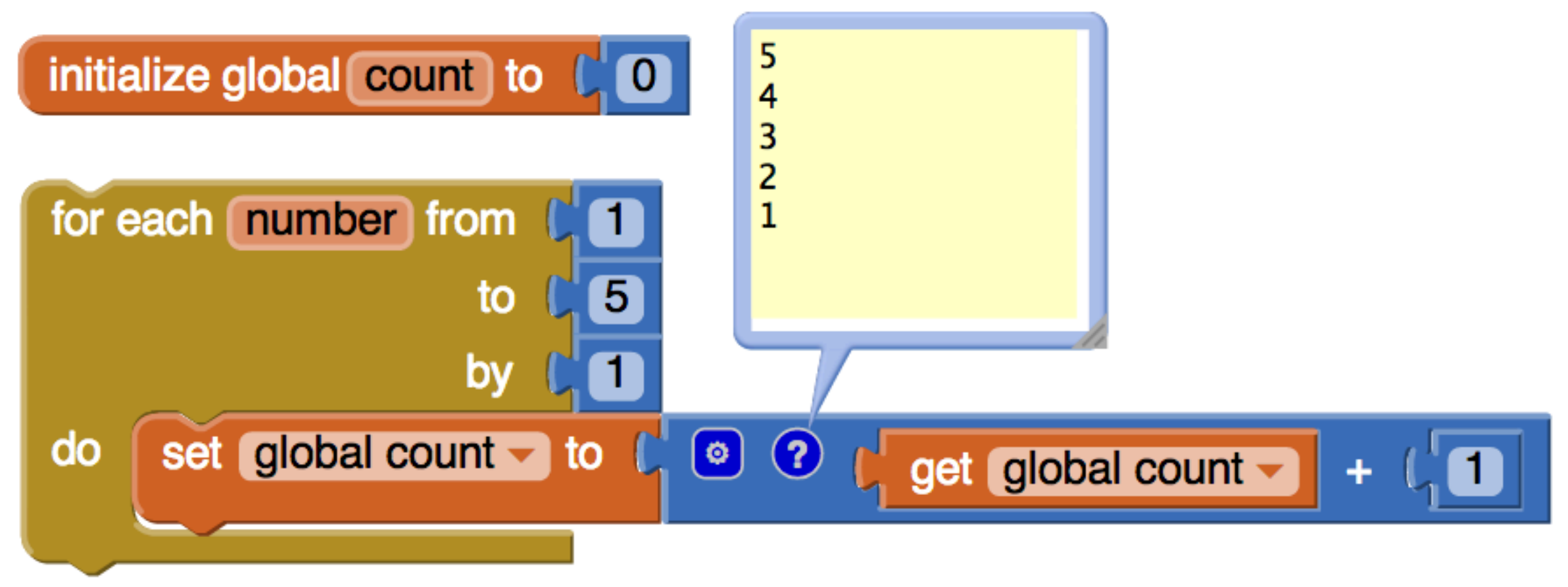
Error from Companion: The operation * cannot accept the arguments: 2 hello

Error from Companion: Select list item: Attempt to get item number 4 of a list of length 0: ()

initialize global name to "hello"
when Button1 .Click
do set Button1 . Text to get global name + 2
when Button2 .Click
do set Button2 . Text to 2 × get global name
when Button3 .Click
do set Button3 . Text to select list item list create empty list index 4
Show Warnings

Watch

When a user puts a "watch" on a block, every time that block is evaluated, the result is sent to the block and displayed sequentially. Users can watch any expression, including variable references and procedure calls that return values.

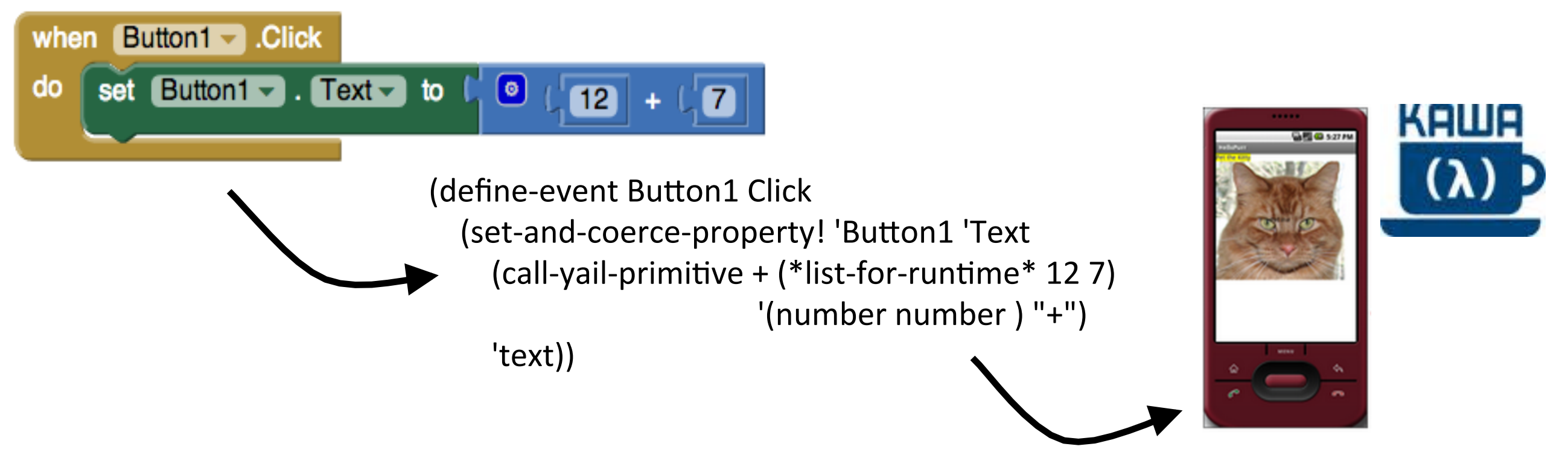

initialize global count to 0
for each number from 1
to 5
by 1
do set global count to get global count + 1
5
4
3
2
1

Data Collection

Currently, recorded error reports include only the error message, the time of the error, and the device generating it. I have been designing the means to trace the errors back to the projects generating them. My design is to store a snapshot of the program at the time of the error in the cloud and send the user name and project ID with the error report to link back to the project. Other work is being done to store the snapshot of the program, but I have been working to store the username and project ID.

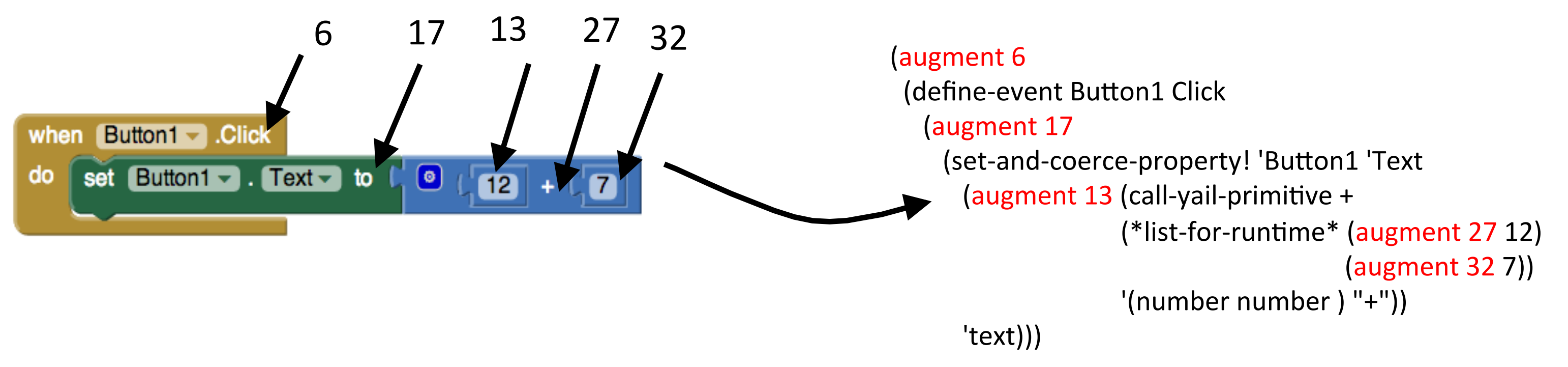
Implementation: Augmented Code

All of my work has been possible because of my implementation of augmented code. Blocks are converted to YAIL (Young Android Intermediate Language) and sent to the device, where a Kawa interpreter runs the program. The previous mechanism for converting blocks to YAIL did not include any information about block identities, so there was no way to associate errors with particular blocks. While the code was evaluated, the device and blocks editor didn't know which block was being executed.



```
(define-event Button1 Click
   (set-and-coerce-property! 'Button1 'Text
      (call-yail-primitive + (*list-for-runtime* 12 7)
                                '(number number ) "+")
      'text))
```

Blocks in an App Inventor have a unique ID within that program. With my implementation, when the blocks are converted to YAIL, the code is wrapped in a tag that has the word "augment" and the block ID. When the device executes that code, it stores the block id as it executes the code from that block.



```
(augment 6
 (define-event Button1 Click
   (augment 17
     (set-and-coerce-property! 'Button1 'Text
       (augment 13 (call-yail-primitive +
                     (*list-for-runtime* (augment 27 12)
                                         (augment 32 7))
                     '(number number ) "+"))
       'text)))
```

Future Work

- The data collected from more detailed error reports could lead to the development of an intelligent tutor that responds when a user encounters a runtime error with suggestions of how to fix it based on what other users did when encountering a similar error.
- A more dynamic watch could be implemented where the value of all expressions are recorded during runtime and then the user has the ability to go back and forth in time to see how the state of the program changed and what was being executed at that point.

Funded by Brachman Hoffman Fund Faculty Small Grants Summer Research Awards and National Science Foundation Grant DUE-1226216. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.